Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.9854883

SEQUENTIAL PATTERN MINING USING APRIORI AND FP GROWTH ALGORITHM

Sujit R Wakchaure, Dr. Rajeev G Vishwakarma
Department of Computer Science and Engineering,
Dr. A. P. J. Abdul Kalam University, Indore (M.P.) - 452016, India
Corresponding Author: sujitw2777@gmail.com

Abstract:
In order to use Apriori, generation of candidate item sets is required. If the itemset in the dataset is particularly vast, the number of instances of these item sets could be rather high. Apriori must perform repeated checks of the database in order to determine whether or not each newly formed itemset is supported, which results in increased expenses. Due to this, it is ineffective when utilised with a significant amount of datasets. For instance, the case in which there is a frequent-one itemset that has 104 elements from the set. The Apriori algorithm must generate more than 107 candidates with a two-length, whose will be assessed and accumulated as a result of their participation in the process. The use of the Apriori technique would be the generation of 2100 probable sets of items with the objective to identify a frequent sequence with size 100 (including v1, v2, v3, v4..., v100). The result would be an illustration of how the process works.

As a result, the yield expenses increase up, and a significant amount of time is lost in the potential candidate generation process (also known as the temporal complexity of the Apriori method). Additionally, as it is trying to enhance the Apriori method in order to verify the numerous candidate itemsets that have been acquired from the numerous sets, it performs a comprehensive search of the database multiple times utilising resources that are quite costly. This, consequently, has an effect on the method whenever there is inadequate storage in the system and a high volume of transactions that occur frequently. Due to this, the technique becomes less effective and more sluggish when applied to more complex huge databases.
The use of FP growth method will allow for the problems of Apriori algorithm to be resolved. As a result, it constitutes advancement over the Apriori technique. There is no requirement for the generation of candidates as a regular pattern is produced automatically. The database is represented by the FP growth method in a version of a tree known as a frequent pattern tree, also known as an FP tree.

In contrast to Apriori, the depth-first search algorithm known as FP-Growth never uses a generate-and-test methodology in its operation. It was initially proposed as a successful approach for mining the entire collection of common patterns. This method makes use of a highly compressed representation of the data that is referred to as an FP-tree [72]. The connection among the sets of objects is preserved by the use of this tree structure. The database is partitioned due to the presence of a single common object. The term "pattern fragment" refers to this portion that has been split up. The item sets contained within these shattered patterns

will then be investigated. Due to this, the time spent investigating common item combinations can be reduced down significantly when utilising this strategy.

Frequent Pattern Tree

The initial itemsets of the database are used to construct the Frequent Pattern Tree, which resembles a tree in its overall layout. The goal of the FP tree is to extract the pattern that occurs most frequently. The FP tree is arranged so that every node stands in for one of the items in the itemset.

The root node is denoted by null value, whereas the item sets are shown by the nodes below it. During the process of generating the tree, the associations between the nodes and the lower nodes, which correspond to the item sets and the other item sets, are kept.

Steps of FP-Tree

The following is an overview of the procedures that were carried out in order to extract the frequent pattern utilising the FP growth technique:

1. The first step of the procedure is to run a search through the database to locate all of the instances of the item sets that are stored there. This stage is identical to the first stage

of the Apriori method. The number of one-item sets that are contained within the database is referred to as the support count or the frequency of one-item sets.

2. Developing the FP tree is the subsequent step in the process. To accomplish this, designing of the base of the tree is needed. The value null is used for representing the root.

3. The next step is to perform another search of the database and investigate the actions taken. Investigate the initial transaction, and determine whatever itemset it contained. The itemset with the highest total count is placed first, followed by the itemset with the subsequent lowest total count, and so on. This indicates that every branch of the tree is built using transactional item sets that are ordered in a decreasing count from highest to lowest.

4. The investigation moves on to the following transaction in the database. The item sets are arranged with the lower counted sets coming first. If each of the item sets associated with this transaction already exist in a separate branch then the prefix attached to this transaction branch will be the same as the one attached to the base.

5. This indicates that the newly created node of the frequent itemset has been connected to the new node of one of the other itemsets in this transaction.

6. In addition to this, the count of the itemset is increased by one each time it is used in the transactions. As new nodes are generated and linked to existing ones in accordance with transactions, the total count of both new and frequent nodes increases by 1.

7. The mining of the newly constructed FP Tree is the subsequent step to be completed. In order to accomplish this, the node with the fewest connections is investigated first, in addition to the lowest node itself and its links. The frequency pattern of size 1 is represented by the node that is the lowest. Then proceed along the route in the FP Tree. One or more of these routes is referred to as a conditional pattern basis.

8. The conditional pattern base constitutes a sub-database that is made up of prefix routes in the FP tree which begin with the suffix at the lowest level of the tree.

9. Develop a Conditional FP Tree, that is created by keeping a count of the item sets along the path. The Conditional FP Tree takes into account the item sets that have support levels that are sufficient enough to achieve the threshold.

10. The Conditional FP Tree is responsible for the generation of Frequent Patterns.

Methodology

An FP-Tree is a type of data structure that is made up of nodes represent different elements, such as a counter, and pathways which indicate different types of transactions. The FP-Growth algorithm relies on two steps, which are as follows:

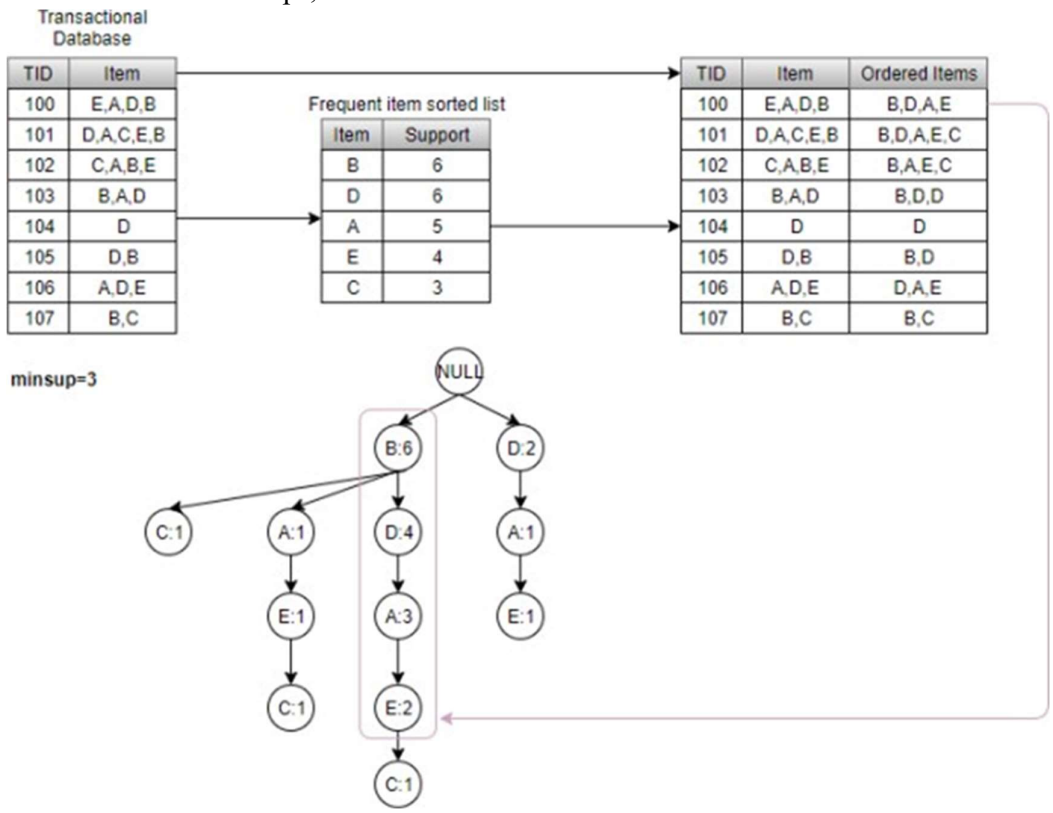

Figure 1: Example of FP-Growth algorithm.

Construction of FP-Tree: To begin the process of constructing the FP-Tree, a single pass through the data is carried out with the aim to generate a support-descending sorted list of frequently ocurring item-sets. After that, it starts with reading every transaction according to the sequence that it appears in the ordered list and mapping this to a path within the FP-Tree as seen in figure 2.2. At this point, the database is only searched over twice: once to determine the number of supporters for every item and, if necessary, prune the resulting nodes, and once more to construct the tree with the nodes' supports ordered from highest to lowest.

Extraction of Frequent Item-sets: The tree is presently being searched in order to locate every item sets for every individual item. In order to accomplish this, discovering a conditional base pattern (CBP) is needed, for every pattern (i.e., prefix-paths within the FP-Tree that are made up of a suffix pattern) [81]. The technique begins by generating a conditional tree based on the CBP, which is then processed in a recursive way. If the dimension of the FP-tree is sufficiently small to be accommodated into main memory, then the extraction of the frequent item-sets entirely from the structure is possible which is kept in memory rather than performing multiple passes across the FP-Tree saved on disc. If the length of the FP-tree is too large to enter into main memory, then the process cannot be performed.

FP-Growth is quicker than Apriori as it preserves complete Apriori data in addition to frequent pattern mining, builds an extremely compact FP-Tree (i.e., overlapping paths) which decreases the expense of checks in subsequent mining techniques, prevents the costly creation of candidates step, and utilises a divide-and-conquer technique to minimise the dimensions of subsequent CPB and conditional FP-Trees [81]. Yet, effectiveness can be significantly impacted when frequent patterns that cannot be stored in memory are present. This is analogous to Apriori. This approach is also neither good for dynamic mining systems (that is, when altering the support threshold in accordance with rules), nor is it acceptable for progressive mining (that is, avoiding resuming mining on the entire database whenever an update takes place).

The iteratively divide and conquer technique is the foundation of the FP-tree method. To begin, a collection of frequent 1-itemsets and their counts are identified. After constructing the CPB with each common pattern as the starting point, then it moves on to constructing the conditional FP-tree of that pattern, referred to as a prefix tree. while the consequent FP-tree is either empty or only comprises a single path, whichever comes first, a single path will result in all the possible combinations of its subpaths, all of which represents a frequent pattern. The items in every transaction are dealt with in the sequence of L.

Flowchart of FP growth Algorithm

The goal of the FP growth method is to generate the frequent patterns, which will allow for the determination of items or subsequent sequences that appear together within the database. Figure 3.2 depicts the flowchart of FP growth algorithm.

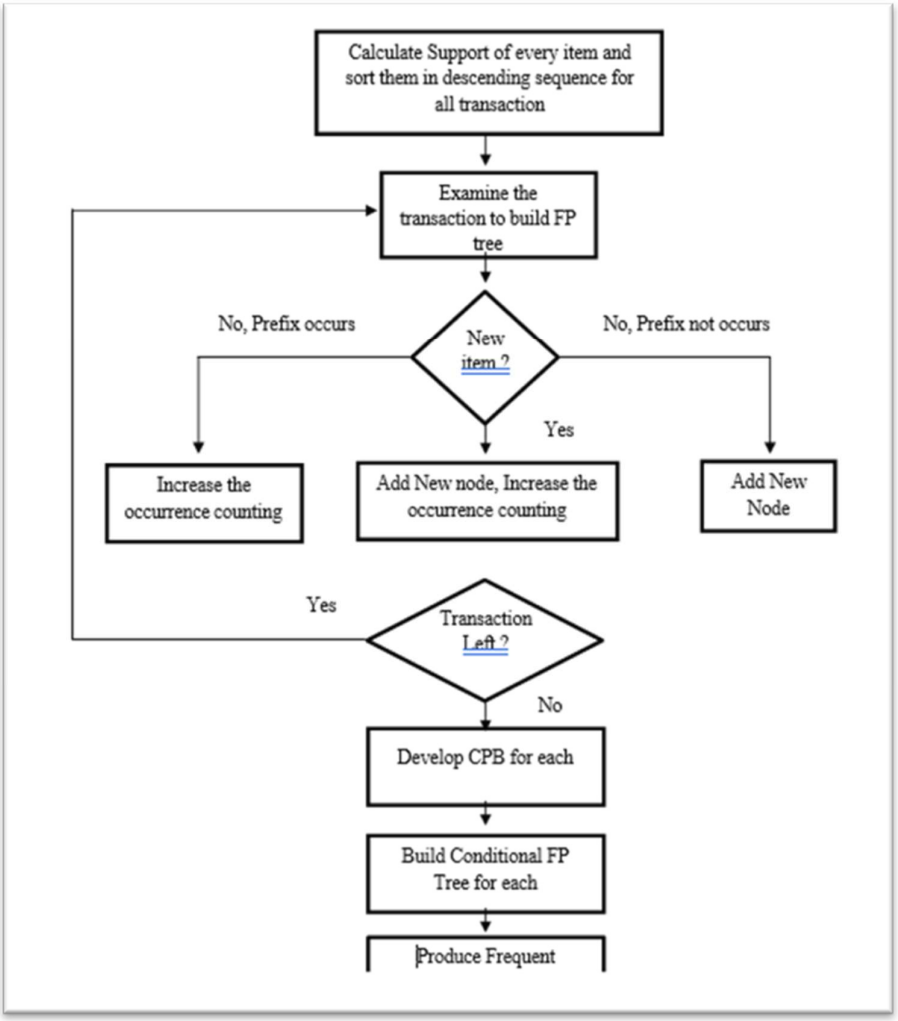

Figure 2 : Flowchart of FP growth Algorithm

The first step is to perform a search through the database to determine the amount of support for all data item that appears in the transactions. After that, a FP tree is built by incorporating all of the elements as nodes and activities as branches. Additionally, the total number of nodes that support specific branches is indicated next to each node. The branches of the tree that correspond to every individual item are then used to construct a secondary database that is referred to as the CPB. It includes the prefix path from the base of the trees all the way up to the item in the tree. All nodes in the route that have a support value that is lower than the minimum support value of 2 are removed, and associated FP trees are built. Thus the pattern are produced. This algorithm's efficiency is superior to that of Apriori as a result of the use of a different technique to locate frequent itemsets with no candidate productions, which cost a lot of storage and computation time respectively. For storing frequent pattern mining into an alternative manner, this technique makes use of a divide-and-conquer technique and a data structure that is referred to as a FP-tree.

The compacting factor of the dataset has an effect on how long it takes for the FP-Growth Method to complete its run [6]. The efficiency of the method will suffer significantly if the conditional FP-Tree of the outcome has numerous branches. This is since the algorithm will need to handle a great deal of conditional FP-Tree. The challenging task of FP-Growth is heavily reliant on the navigating in FP-Tree for every single item in the header of the table, and this in turn is determined by the depth of the tree. For any conditional FP-Tree, the deepest level of a tree is limited by a parameter denoted by d. If this is the case, the degree of complexity of the method is denoted by the notation O (the number of distinct items in the header table * greatest depth the tree can reach) = O (d*d).
The FP-Growth method is more difficult to comprehend and put into practise than the Apriori method. On the other hand, this strategy utilises resources in a more effective manner. As a result, the FP-Growth methodology has a significant amount of potential improvement and enhancement.

Algorithm

Method FP-Tree (T, α)
{
If Tree T comprises a single route R
Then
For every β = nodes in R do
Pat = β U α
Support S = minimum(S of the nodes in β);
Else
For every α in the T's header do
Pat β = αi U α;
With s = αi, s;
Build CPB of β
T = build conditional FP-tree of β
If T! = ∅ then
Call FP-Tree (T, β)
}

Advantages Of FP Growth Algorithm

- Unlike Apriori, which searches the transactions during every iteration, this method only has to do a database scanning twice. Apriori analyses the database during every iteration.
- The matching of items is skipped in this approach, which results in a significant increase in processing speed.
- A condensed version of the database is kept in the memory.
- Extracting both short and lengthy frequent patterns can be done in an efficient and extensible manner using it.

Disadvantages Of FP-Growth Algorithm

- FP Tree is more complicated and time-consuming to construct than Apriori.
- It is probable to be more expensive.
- If the database is quite huge, there is a possibility that the method will not fit into the shared memory space.

a. Experimental findings and Discussion

Experiments are carried out on a personal computer that features a processor 2.60 GHz belonging to the Intel(R) Core(TM) 8th gen intel-3230 CPU, 4 gigabytes of primary memory, and the Microsoft Windows 7 operating system. The experiment utilizes the Adult dataset which is taken from the UCI Machine Learning Repository.

Consider table 1 and figure 3 which illustrate the performance assessment of the Conventional Apriori and FP Growth approach based on the total number of rules and the user- specified minimal confidence cut off.

Table 1: No of Rules verses User specified minimum confidence cut off (Apriori and FP Growth Algorithm)

Minimum cutoff	FP	APRIORI
0.1	273	261
0.15	268	250
0.2	262	248
0.25	286	249
0.3	197	162
0.35	165	148

The minimum confidence level that the user chooses to display is indicated along the X-axis, and the total amount of rules is displayed along the Y-axis. According to the information that can be observed in figure 3, the number of rules will decrease in a linear way whenever the user-specified value for the lowest level of confidence cutoff increases. When the support value is increased, the number of rules that are generated by the FP Growth algorithm is greater than that generated by the Apriori algorithm. Therefore, the FP growth algorithm is superior to the apriori algorithm in terms of performance.

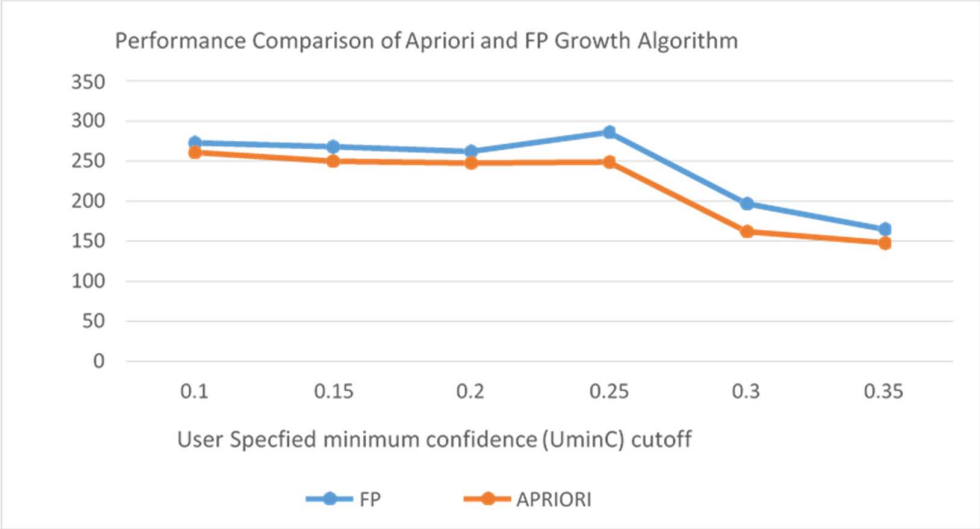

Figure 3: No of Rules for different user specified minimum confidence cut off (Apriori and FP Growth Algorithm)

Consider the following table 2 and figure 4, which present an evaluation of the performance of the Apriori and FP growth method based on the amount of time it takes to execute the algorithm along with the various support values it uses. The amount of time necessary to mine the frequent objects is referred to as the execution time.

Table 2: Execution Time verses Support Value (Apriori and FP Growth Algorithm)

Minimum cutoff	FP	APRIORI
0.1	272	289
0.2	268	278
0.3	254	264
0.4	244	254
0.5	189	207
0.6	159	188

On the graph, the X-axis represents the various support values, and the Y-axis represents the amount of time it took to complete the task. Figure 4.3 reveals that when the support value is

raised, the execution time of the Apriori and FP growth algorithm decreases in a pattern that is linear. When the support value is increased, the amount of time needed to execute the FP Growth algorithm is less than that time required by the Apriori algorithm. Therefore, the FP growth algorithm performs better than to the Apriori algorithm in terms of execution time.

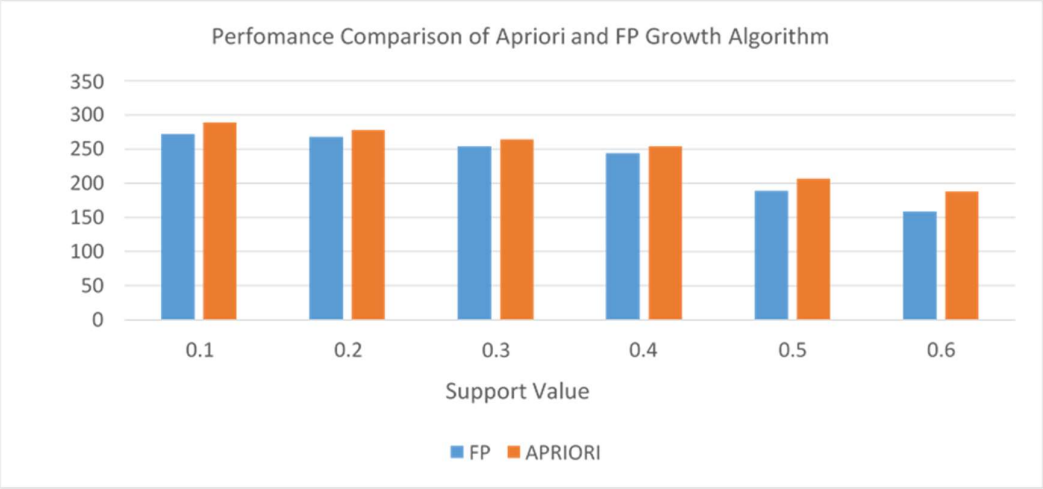

Figure 4: Execution Time for different Support Value (Apriori and FP Growth Algorithm)

Consider the following table 3 and figure 5, which demonstrate an assessment of the performance of the Apriori and FP growth method based on the amount of time it takes to execute and the number of transactions it processes.

Table 3: Execution Time verses no of transaction (Apriori and FP Growth Algorithm)

Number of Transaction	Apriori	FP
500	15	11
1000	28	24
1500	42	35
2000	55	43
2500	71	62
3000	89	83

The Y-axis displays the amount of time it took to complete each transaction, while the X-axis depicts the total number of transactions. Figure 4.5 reveals that the execution time of the Apriori and FP growth method will automatically rise in a linear form whenever the number of transactions is increased. When the number of transactions is increased, the amount of time needed to execute the FP Growth algorithm is less than that time required by the Apriori algorithm. Therefore, the FP growth algorithm performs better than to the apriori algorithm in terms of execution time.

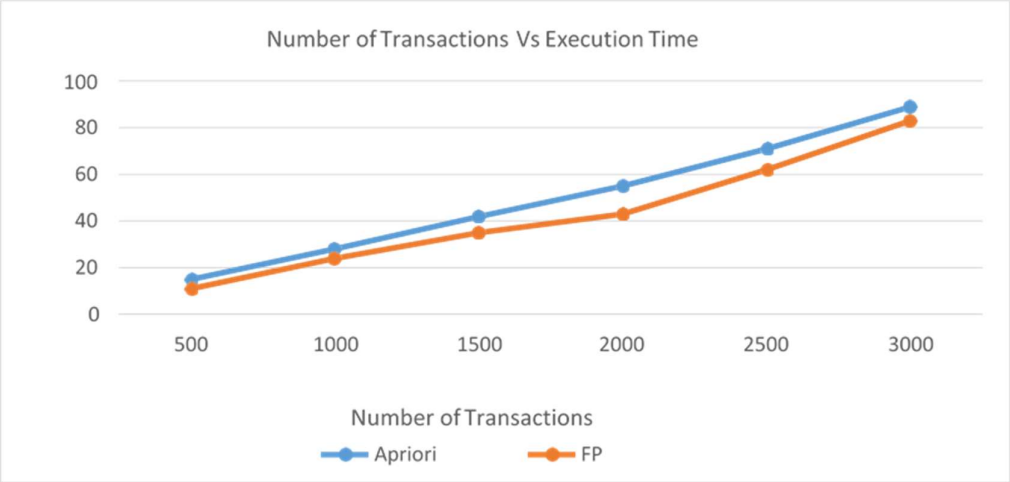

Figure 5: Execution Time for various no of transaction (Apriori and FP Growth Algorithm)

Consider the following table 4 and figure 6, which show an examination of the effectiveness of the Apriori and FP growth algorithm based on the amount of time it takes to execute and the number of transactions it processes.

Table 4: Memory Usage verses different support Value (Apriori and FP Growth Algorithm)

Support Value	FP	APRIORI
0.1	250	282
0.2	240	270
0.3	230	257
0.4	222	247
0.5	185	200
0.6	167	180

Along the X-axis is a representation of the various support values, and along the Y-axis is the amount of memory space that is being used. It can be seen in figure 4.6 that the memory use of the Apriori and FP growth method gradually grows in a monotonous way if the support value is raised. When the support value is increased, the amount of memory space needed by the FP Growth algorithm is less than that of memory space required by the Apriori algorithm. Therefore, the FP growth algorithm performs better than to the apriori algorithm in terms of memory space.

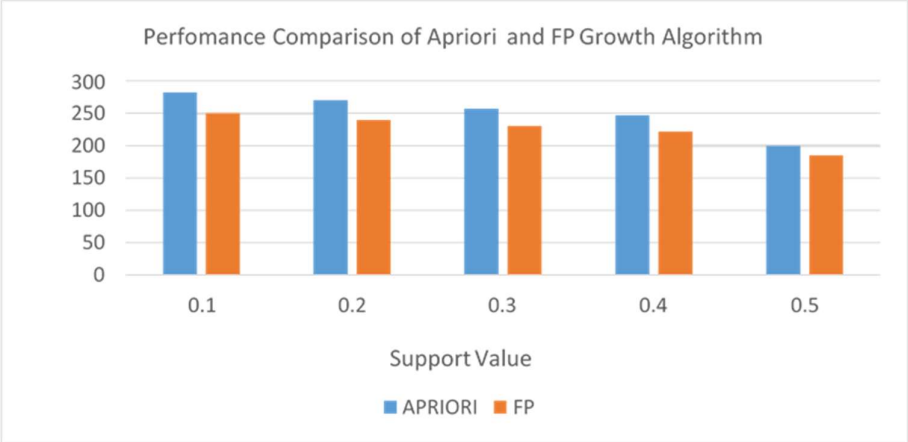

Figure 6: Memory Usage for different support value (Apriori and FP Growth Algorithm)

Conclusion:

In this investigation, a comparative analysis of conventional Apriori algorithm and FP Growth algorithm is performed. The methods, benefits and drawbacks of both techniques are discussed in detail. Both the methods effectively mine the frequent occurring patterns from database. The FP Growth algorithm finds the most frequently used item sets with no need for candidate item set generation, in contrast to the Apriori method, which finds the frequent item sets via generating candidate item sets. Various experiments are performed using conventional Apriori and FP growth techniques with regards to aspects: Total number of transaction, memory consumption, and execution time. It is observed that the execution time and memory consumption of FP growth technique is less as compared to conventional Apriori technique using different support values. But, there is a need to propose an enhanced approach which will enhance the performance of conventional Apriori and FP growth technique over huge large pattern of datasets. Also, there is need to improve the computational time and memory utilization of conventional strategy.

References

[1] Q. Xuan, H. Jiuyuan and W. Weitao, "Research on Improvement of Parallel Apriori Algorithm Based on Boolean Matrix and Weight," 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA), Xiangtan, China, 2019, pp. 96-99, doi: 10.1109/ICICTA49267.2019.00027.

[2] Laurentinus O. Rizan, Sarwindah, Hamidah, R. Sulaiman and Hengki, "Data Mining Using Apriori Algorithm and Linear Regression in Product Recommendations," 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2021, pp. 217-223, doi: 10.1109/ISRITI54043.2021.9702791.

[3] P. T. Munukuntla, C. K. Somalraju, K. V. Gangisetty, S. K. Satapathy, S. Mishra and P. K. Mallick, "Grocery Recommendation using Improvised Apriori Algorithm," 2022 OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 2022, pp. 1-5, doi: 10.1109/OCIT56763.2022.00015.

[4] A. Sani, Samuel, N. N. P, B. Waseso, G. Gunadi and T. Haryanto, "Data Mining on Sales Transaction Data Using the Association Method with Apriori Algorithm," 2022 10th International Conference on Cyber and IT Service Management (CITSM), Yogyakarta, Indonesia, 2022, pp. 1-5, doi: 10.1109/CITSM56380.2022.9935890.

[5] I. M. Stubarev, O. K. Alsowa and A. A. Yakimenko, "Effectiveness Research of the Apriori Algorithm Implementations as Part of the Recommendation System," 2021 XV International Scientific-Technical Conference on Actual Problems Of Electronic Instrument Engineering (APEIE), Novosibirsk, Russian Federation, 2021, pp. 587-590, doi: 10.1109/APEIE52976.2021.9647623.

[6] M. Javad Shayegan and P. Asgari Namin, "An Approach to Improve Apriori Algorithm for Extraction of Frequent Itemsets," 2021 7th International Conference on Web Research (ICWR), Tehran, Iran, 2021, pp. 206-210, doi: 10.1109/ICWR51868.2021.9443137.

[7] H. Wang, H. Jiang, H. Wang and L. Yuan, "Research on an improved algorithm of Apriori based on Hadoop," 2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS), Xi'an, China, 2020, pp. 242-245, doi: 10.1109/ISPDS51347.2020.00057.

[8] N. K. Verma, S. Kumar, M. Kumar and P. Saxena, "An Alternate Approach to Improve Access Time for Defining Frequent Item Set Through 'A-Apriori' in Textual Data Set," 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, 2020, pp. 1-6, doi: 10.1109/ICRAIE51050.2020.9358344.

[9] R. Sun, "A Combination of Prefixed-Itemset and Database Optimization to Improve Apriori Algorithm on Hadoop Cluster," 2019 4th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), Hohhot, China, 2019, pp. 992-9925, doi: 10.1109/ICMCCE48743.2019.00223.

[10] N. Karimtabar and M. J. Shayegan Fard, "An Extension of the Apriori Algorithm for Finding Frequent Items," 2020 6th International Conference on Web Research (ICWR), Tehran, Iran, 2020, pp. 330-334, doi: 10.1109/ICWR49608.2020.9122282.

[11] M. N. M. Br.Sipahutar, O. S. Sitompul and Sutarman, "Association Rules Analysis Using Algorithm Apriori And Fuzzy Normalization," 2021 IEEE International Conference on Health, Instrumentation & Measurement, and Natural Sciences (InHeNce), Medan, Indonesia, 2021, pp. 1-5, doi: 10.1109/InHeNce52833.2021.9537210.

[12] Z. Mar and K. K. Oo, "An Improvement of Apriori Mining Algorithm using Linked List Based Hash Table," 2020 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 2020, pp. 165-169, doi: 10.1109/ICAIT51105.2020.9261804.

[13] X. Ren, "Application of Apriori Association Rules Algorithm to Data Mining Technology to Mining E-commerce Potential Customers," 2021 International Wireless Communications and Mobile Computing (IWCMC), Harbin City, China, 2021, pp. 1193-1196, doi: 10.1109/IWCMC51323.2021.9498773.